IEEE SA
STANDARDS ASSOCIATION

IEEE Standard for Technical Requirements and Evaluating Knowledge Graphs

STANDARDS

IEEE Computer Society

Developed by the
Knowledge Engineering Standards Committee

IEEE Std 2807.1™-2024

IEEE

IEEE Standard for Technical Requirements and Evaluating Knowledge Graphs

Developed by the

Knowledge Engineering Standards Committee
of the
IEEE Computer Society

Approved 6 June 2024

IEEE SA Standards Board

Abstract: This standard defines technical requirements, performance metrics, evaluation criteria and test cases for knowledge graphs. The mandatory test cases include data input, metadata, data extraction, data fusion, data storage and retrieval, inference and analysis, and knowledge graph display. This standard can be applied in various organizations that plan, design, develop, implement, and apply knowledge and in organizations that develop support technologies, tools, and services to knowledge graphs.

Keywords: IEEE 2807.1™, knowledge graph, knowledge graph application, knowledge graph construction, knowledge graph evaluation, performance metrics

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 979-8-8557-1142-4 STD27267
Print: ISBN 979-8-8557-1143-1 STDPD27267

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (https://standards.ieee.org/ipr/disclaimers.html), appear in all IEEE standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers involved in technical working groups are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning all standards, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement IEEE Standards documents do not guarantee safety, security, health, or environmental protection, or compliance with law, or guarantee against interference with or from other devices or networks. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document should rely upon their own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus balloting process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English language version published by IEEE is the approved IEEE standard.

Use by artificial intelligence systems

In no event shall material in any IEEE Standards documents be used for the purpose of creating, training, enhancing, developing, maintaining, or contributing to any artificial intelligence systems without the express, written consent of IEEE SA in advance. "Artificial intelligence" refers to any software, application, or other system that uses artificial intelligence, machine learning, or similar technologies, to analyze, train, process, or generate content. Requests for consent can be submitted using the Contact Us form.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual is not, and shall not be considered or inferred to be, the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE or IEEE SA. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s). News releases about IEEE standards issued by entities other than IEEE SA should be considered the view of the entity issuing the release rather than the formal position of IEEE or IEEE SA.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents**.

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments or questions, except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE SA working group. You can indicate interest in a working group using the Interests tab in the Manage Profile and Interests area of the IEEE SA myProject system.[1] An IEEE Account is needed to access the application.

Comments on standards should be submitted using the Contact Us form.[2]

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory

[1]Available at: https://development.standards.ieee.org/myproject-web/public/view.html#landing.
[2]Available at: https://standards.ieee.org/about/contact/.

requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; https://www.copyright.com/. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore or contact IEEE.[3] For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website.[4] Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional

[3]Available at: https://ieeexplore.ieee.org/browse/standards/collection/ieee.
[4]Available at: https://standards.ieee.org/standard/index.html.

Resources Details section. Errata are also available in IEEE Xplore. Users are encouraged to periodically check for errata.

Patents

IEEE standards are developed in compliance with the IEEE SA Patent Policy.[5]

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. IEEE standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, data privacy, and interference protection practices and all applicable laws and regulations.

[5]Available at: https://standards.ieee.org/about/sasb/patcom/materials.html.

Participants

At the time this standard was completed, the Knowledge Graph Evaluation Working Group had the following membership:

Ruiqi Li, *Chair*
Lun Li, *Vice Chair*
Fan Yang, *Secretary*
Qi Jia, *Editor*

Organization Represented	*Name of Representative*
Alipay (China) Technology Co., Ltd.	Zhihui Guo
Anhui University	Shu Zhao
Beijing Haizhi Technology Group Co., Ltd	Siyu Li
Beijing PERCENT Technology Group Co., Ltd.	Yijing Liu
CESI (Guangzhou) Standards & Testing Institute	Mingying Zhang
CETC Big Data Research Institute Co., Ltd.	Yang Cao
CETC LES Information System Group Co., Ltd.	Yan Hong
Chengdu University of Information Technology	Fan Yang
China Electric Power Research Institute (State Grid Corporation of China)	Zhenyuan Ma
China Electronics Standardization Institute	Nan Guo
China Resources Digital Co., Ltd	Xing Zhang
China Resources Intelligent Computing Technology（Guangdong）Co., Ltd.	Minsen Yao
China Southern Power Grid Co., Ltd.（EHV Power Transmission Company）	Qiang Li
Chongqing Changan Automobile Co., Ltd.	Yonggang Luo
Hisense Group Holdings Co., Ltd.	Wei Liu
Huawei Technologies Co., Ltd.	Weijian Sun
IEIT Systems Co., Ltd	Rengang Li
Institute of Biomedical Engineering, Chinese Academy of Medical Sciences & PUMC	Jiangbo Pu
Jiaxinda Private Fund Management Co., Ltd.	Kaiqi Wu
Midea Group	Yasen Cai
Nanjing University of Aeronautics and Astronautics	Fuhui Zhou
NARI Group Corporation (State Grid Corporation Of China)	Jingyi Su
Neusoft Research of Intelligent Healthcare Technology, Co., Ltd. (Neusoft Corporation)	Weiguang Wang
Peking University	Lei Zou
Shanghai Artificial Intelligence Industry Association	Xi Chen
Shanghai Jiao Tong University	Luoyi Fu
Shanghai University of Engineering Science	Juan Zhang
Shenzhen CESI Information Technology Co., Ltd.	Lun Li
Shenzhen Star Innovation Digital Economy Research Center	Ruihu Tan
Shenzhen Zhezhi Huilian Techonogy Co., Ltd.	Xueqin Zhang

Sichuan University ..... Jiancheng Lv

Siheria Technologies Co., Ltd. ..... Pengda Hong

Tianfu (Dongguan) Standards Technology Co., Ltd. ..... Cheng Zheng

Tianjin University ..... Yanfei Liu

Tongling University ..... Feng Qian

Transwarp Technology(Shanghai) Co., Ltd. ..... Ren Yi

Unilumin Group ..... Yongheng Huang

Unionman Technology Co., Ltd. ..... Haipeng Zhao

Wiseweb Technology Group Co., Ltd ..... Chengbin Jia

Xi'an Jiaotong University ..... Jie Ma

Zhejiang Createlink Technology Co., Ltd. ..... Zhou Yan

The Working Group gratefully acknowledges the contributions of the following participants. Without their assistance and dedication, this standard would not have been completed.

Qing Ai
Wei Cai
Jie Chen
Qian Chen
Zhang Chen
Yuhang Cheng
Feihu Duan
Baoyu Fan
Zhijun Fang
Lixin Gao
Li Han
Baoxuan Hong
Wanfu Hong
Hou Hongyi
Lin Hu
Shudong Huang
Shiqi Jia
Guotao Jiao
Jia Li
Ting Li
Xuemei Li
Lei Liang
Dieshou Lin
Bowen Liu
Changyu Liu
Qian Mo
Xiaofeng Mou
Zhang Nan
Jim Peng
Ma Rui
Bozhong Shao
Zhai Shidan
Jianfei Tang
Yifan Tang
Jie Tong
Qing Tong
Pengjiang Wan
Pinghui Wang
Yihao Wang
Qihui Wu
Zhengxun Xia
Lecheng Xie
Ziqi Xiong
Yankai Xue
Bo Yao
Qingyun Yin
Shanqin Yu
Tian Yu
Lu Yuan
Fatao Zhang
Xie Zheyu
Li Zijian

The following members of the entity Standards Association balloting group voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Alipay (China) Technology Co. Ltd.
Anhui University
Beijing Haizhi Technology Group Co., Ltd
Beijing Knowledge Atlas Technology Co., Ltd
CETC Big Data Research Institute Co., Ltd
China Electronic Standardization Institute
China Resources Digital Co., Ltd.
Chongqing Changan Automobile Co., Ltd.
CloudWalk Technology
CRRC Zhuzhou Institute Co., Ltd
Dell Inc.
Hubbell
IEIT Systems Co., Ltd.
Institute of Biomedical Engineering
Intel
Midea Group
Nanjing Research Institute of Electronics Engineering
Neusoft Corporation
Peking University
Shanghai Artificial Intelligence Industry Association
Shanghai University of Engineering Science
Siheria Technologies Co., Ltd.
State Grid Corporation of China (SGCC)
Technology(Shanghai) Co., Ltd
Transwarp
University of South China
Wiseweb Technology Group Co., Ltd.
Xi'an Jiaotong University
Zhejiang Createlink Technology Co., Ltd

When the IEEE SA Standards Board approved this standard on 6 June 2024, it had the following membership:

David J. Law, *Chair*
Jon Walter Rosdahl, *Vice Chair*
Gary Hoffman, *Past Chair*
Alpesh Shah, *Secretary*

Sara R. Biyabani
Ted Burse
Stephen Dukes
Doug Edwards
J. Travis Griffith
Guido R. Hiertz
Ronald W. Hotchkiss
Hao Hu
Yousef Kimiagar
Joseph L. Koepfinger*
Howard Li
Xiaohui Liu
John Haiying Lu
Kevin W. Lu
Hiroshi Mano
Paul Nikolich
Robby Robson
Lei Wang
F. Keith Waters
Sha Wei
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 2807.1-2024, IEEE Standard for Technical Requirements and Evaluating Knowledge Graphs.

Artificial intelligence (AI) applications, including machine translation, speech recognition, image classification, and information retrieval, have achieved remarkable success. AI applications are developing toward expertise AI, like medical diagnosis, autonomous vehicles with AI, and smart factories. Knowledge graphs can assist in linking machine learning experts with experts from different industries, for example, medical care and transportation, and further transform human knowledge into machine knowledge. Knowledge graphs provide a natural and efficient way to represent entities and their relationships in a format that is understandable by machines and humans.

However, knowledge graphs meet a critical need to many enterprises and organizations by providing the structured data and factual knowledge that drives product development. This standard addresses the customer need for a specific and objective way to evaluate the performance of knowledge graphs. This standard can promote the stability, availability, and usability of knowledge graphs, thus lowering the barriers for model selection by users and producers.

Contents

IEEE Standard for Technical Requirements and Evaluating Knowledge Graphs

1. Overview

1.1 Scope

This standard defines technical requirements, performance metrics, evaluation criteria and test cases for knowledge graphs. The mandatory test cases include data input, metadata, data extraction, data fusion, data storage and retrieval, inference and analysis, and knowledge graph display.

1.2 Purpose

The purpose of the standard is to promote the stability, availability, and usability of knowledge graphs, thus lowering the barrier for model selection for users and producers.

1.3 Word usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).[6,7]

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

[6]The use of the word *must* is deprecated and cannot be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.
[7]The use of *will* is deprecated and cannot be used when stating mandatory requirements; *will* is only used in statements of fact.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 2807™-2022, IEEE Standard for Framework of Knowledge Graphs.[8,9]

IEEE Std 7002™-2022, IEEE Standard for Data Privacy Process.

ISO/IEC/IEEE 29119-1:2021, Software and systems engineering—Software testing—Part 1: General concepts.

3. Acronyms and abbreviations

API	application programming interface
CSV	comma-separated values
JSON	JavaScript object notation
OWL	Web Ontology Language
QPS	queries per second
RDF	resource description framework
SDK	software development kit
XML	extensible markup language

4. Quality evaluation framework

A knowledge graph is a structured form of collection including knowledge elements and their connections description. According to the knowledge graph technical framework proposed by IEEE Std 2807-2022 and general concepts of software testing proposed by ISO/IEC/IEEE 29119-1:2021,[10] the knowledge graph construction and application system can be divided into the following functional modules for construction of knowledge graphs:

- Knowledge representation
- Knowledge modeling
- Knowledge acquisition
- Knowledge storage
- Knowledge fusion
- Knowledge computing
- Knowledge provenance
- Knowledge evolution

[8]The IEEE standards or products referred to in this clause are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.
[9]IEEE publications are available from The Institute of Electrical and Electronics Engineers (https://standards.ieee.org/).
[10]Information on references can be found in Clause 2.

— Knowledge visualization

— Other related modules

The can also be divided into the following functional modules for application of knowledge graphs:

— Knowledge services

— Knowledge maintenance

— System management

— Other related modules

Accordingly, the quality evaluation framework of knowledge graph construction and application systems is shown in Figure 1 and Figure 2, which can be divided into the following:

a) Quality evaluation framework for modules related to knowledge graph construction: This focuses on the level of support for various activities during the knowledge graph construction process.

b) Quality evaluation framework for modules related to knowledge graph applications: This primarily focuses on ensuring the system features during the knowledge graph application process, including the following:

    1) Responsiveness: This measure evaluates the system and the accompanying knowledge graph's ability to provide prompt and accurate responses to external input requests or operations. This characteristic can be subdivided into sub-features such as knowledge graph scale, knowledge quality, response efficiency, response accuracy, real-time response, and more.

    2) Friendliness: The measure assesses the ease of use, understanding, and learnability of the system and the accompanying knowledge graph. This characteristic can be subdivided into sub-features such as usability, inheritability, maintainability, and more.

    3) Reliability: This measure evaluates the system's ability to maintain its availability in the face of external disruptions. This characteristic can be subdivided into sub-features such as availability, maturity, fault tolerance, ease of recovery, and more.

    4) Portability: This measure assesses the system and the accompanying knowledge graph's ability to run reliably across various software and hardware systems. This characteristic can be subdivided into sub-features such as compatibility, scalability, and more.

    5) Security: This measure evaluates the system and the accompanying knowledge graph's ability to avoid unauthorized access, use, theft, destruction, alteration, forgery, and other potential threats to security. This characteristic can be subdivided into sub-features such as security, confidentiality, controllability, privacy, auditability, and more.

    6) Scenario support: This measure assesses the system and the accompanying knowledge graph's capability to support and implement knowledge graph application scenarios across various industries and specific industry-specific use cases.

This framework focuses on key metrics related to knowledge graph construction and knowledge graph application. Some of the basic, general software-related quality evaluation metrics and measurement methods can be referred to in the standards for systems and software engineering. The requirements on data privacy can refer to IEEE Std 7002-2022.

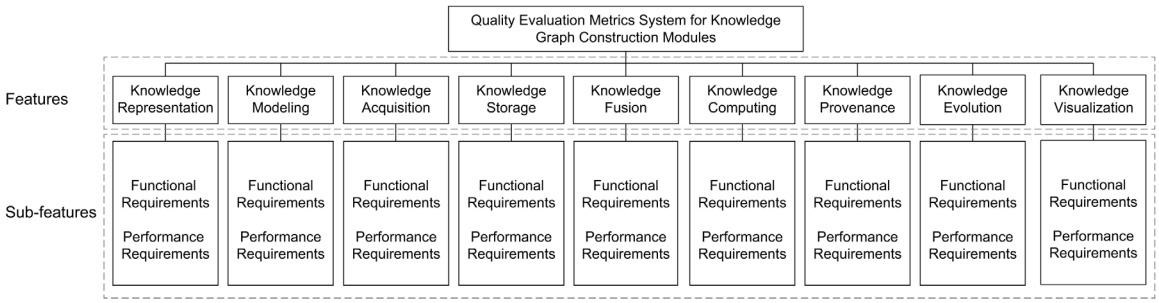

Figure 1—Quality evaluation indicator system for modules related to knowledge graph construction

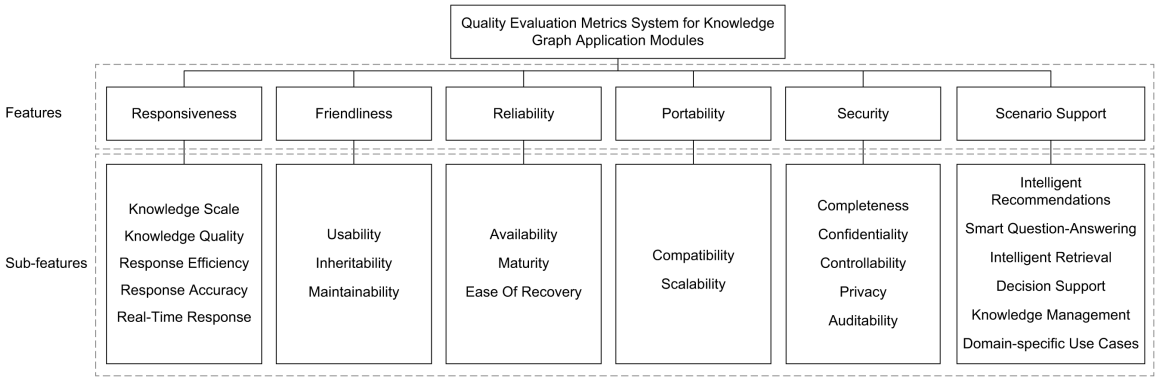

Figure 2—Quality evaluation indicator system for modules related to knowledge graph applications

5. Quality requirements for related modules of knowledge graph construction

5.1 Knowledge representation

5.1.1 Functional requirements

The functional requirements of knowledge representation related modules include, but are not limited to, the following:

a) It shall support symbolic representation of knowledge, such as resource description framework (RDF), Web Ontology Language (OWL), rule language, etc.

b) It shall support numerical representation of knowledge, such as vectors (including embedding and other forms), tensors, etc.

c) It shall support the definition of frameworks, semantic web, ontology, XML, and logical systems, such as first-order logic, higher-order logic, etc.

d) It shall support the representation of axioms, rules, constraints, etc.

e) It shall support the setting, editing, and modification of namespaces.

f) It should support description logic, terminology knowledge, assertion knowledge, etc.

5.1.2 Performance requirements

Performance testing metrics for knowledge representation related modules shall include, but are not limited to, the following:

a) The expressive ability of knowledge representation models, which is to measure the generalization ability of knowledge representation models on multiple tasks, can be achieved through sampling testing. The expressive ability of knowledge representation models is calculated in Equation (1).

$$X_{R\_P} = \frac{A_{R\_P}}{B_{R\_P}} \tag{1}$$

where

A_{R_P} is the number of representation tasks that the knowledge representation model can achieve
B_{R_P} is the total number of representation tasks

b) The computational efficiency of knowledge representation models, which is to measure the time efficiency of the model in generating knowledge representations in specific application scenarios, can be achieved through sampling testing. The computational efficiency of knowledge representation models is calculated in Equation (2).

$$X_{R\_P} = \frac{T_{R\_P}}{B_{R\_P}} \tag{2}$$

where

T_{R_P} is the total time to generate knowledge representations through the knowledge representation model
B_{R_P} is the total number of knowledge representations which generated by the knowledge representation model

5.2 Knowledge modeling

5.2.1 Functional requirements

The functional requirements of knowledge modeling related modules include, but are not limited to, the following:

a) It shall support the definition of entity types, relationships, attributes, and events in the ontology model.

b) It shall support the definition of ontology model constraints, such as upper and lower levels, range of values, and other constraints.

c) It shall support manual ontology modeling.

d) It shall support visualization of ontology models.

e) It shall support manual editing of concepts, relationships, attributes, and other functions.

f) It shall support the import and export of ontology models and schemas.

g) It should support semi-automated ontology modeling.

h) It should support the definition of logical systems.

i) It shall support multiple presentation languages and storage formats.

j) It shall support the merging of ontologies.

5.2.2 Performance requirements

Performance testing metrics for knowledge modeling related modules shall include, but are not limited to, the following:

a) The semantic clarity of the ontology model, which is to measure whether the constructed ontology model provides clear and objective semantic definitions for the terms contained in it, shall be measured through sampling testing. Semantic clarity of ontology model is calculated in Equation (3).

$$X_{O\_C} = \frac{A_{O\_C}}{B_{O\_C}} \tag{3}$$

where

A_{O_C} is the number of entity types and relationship types with semantic description or definition
B_{O_C} is the total number of entity types and relationship types in the ontology model

b) The completeness of the ontology model, which is to measure whether the constructed ontology model fully expresses the meaning of the terms in the described field, shall be measured through sampling testing. The completeness of the ontology model is calculated in Equation (4).

$$X_{O\_I} = \frac{A_{O\_I}}{B_{O\_I}} \tag{4}$$

where

A_{O_I} is the number of entity types and relationship types that reach the completeness requirements
B_{O_I} is the total number of entity types and relationship types in the ontology model

c) The consistency of the ontology model, which is to measure whether the constructed ontology model can correctly and consistently present objects and information, whether there are conflicts in the definitions of entity types, relationship types, attributes, etc., in the constructed ontology model, and whether the reasoning or calculation results derived from the terms do not conflict with the meaning of the terms themselves, can be achieved through sampling testing. Consistency of ontology model is expressed in Equation (5).

$$X_{O\_U} = \frac{A_{O\_U}}{B_{O\_U}} \tag{5}$$

where

A_{O_U} is the number of entity types and relationship types that meet consistency requirements
B_{O_U} is the total number of entity types and relationship types in the ontology model

d) The maximum monotonicity of the ontology model, which is to measure whether there are repeat or overlapping definitions of terms in the ontology model, can be achieved through sampling testing. The maximum monotonicity of the ontology model is calculated in Equation (6).

$$X_{O\_M} = \frac{A_{O\_M}}{B_{O\_M}} \tag{6}$$

where

A_{O_M} is the number of entity types and relationship types that have definition repeat or overlapping
B_{O_M} is the total number of entity types and relationship types in the ontology model

e) The generalizability of the ontology models, which is to measure the semi-automatic ontology modeling ability of ontology models, whether they can be applied to new tasks and data sets without

the need for tables or with only minor modifications, can be achieved through sampling testing. Generalizability of ontology models is calculated in Equation (7).

$$X_{O\_T} = \frac{A_{O\_T}}{B_{O\_T}} \tag{7}$$

where

A_{O_T} is the number of entity types and relationship types that reach the generalization criteria for performance

B_{O_T} is the total number of entity types and relationship types in the ontology model

f) The compatibility and scalability of the ontology models, which is to measure whether the designed ontology model can achieve compatibility or inheritance with existing ontology models, and whether ontology models in the same tool can be extended, such as:

1) The designed ontology model shall support compatibility with other standard ontology models in the field.

2) The designed ontology model shall support the inheritance of existing entity type systems.

3) The designed ontology model shall support the adjustment of entity type attributes after changes or updates to business or input data sources.

4) Compatibility shall be measured through sampling testing. Compatibility is calculated in Equation (8).

$$X_{O\_S}\frac{A_{O\_S}}{B_{O\_S}} \tag{8}$$

where

A_{O_S} is the number of entity types and relationship types that meet compatibility requirements within the intended compatibility interface

B_{O_S} is the total number of entity types and relationship types in the intended compatibility interface

g) The reusability of the ontology models, which is to measure whether the designed ontology model should support reuse between multiple knowledge graphs.

5.3 Knowledge acquisition

5.3.1 Functional requirements

The functional requirements of knowledge acquisition related modules include, but are not limited to, the following:

a) It shall support the import of data in multiple formats, such as CSV, JavaScript object notation (JSON), extensible markup language (XML), PDF, triples, etc.

b) It shall support entity extraction, relationship extraction, attribute extraction, etc., for structured and semi-structured data.

c) It shall support the definition of extraction rules for structured, semi-structured, and unstructured data, such as:

1) For structured data, using mapping rules, regular expressions, model extraction, etc.

2) For semi-structured data, using rule functions, wrapper induction, etc.

3) For unstructured data, using machine learning models, custom matching templates, etc.

d) It shall support entity extraction, relationship extraction, attribute extraction, metadata extraction, etc., for unstructured data such as text, images, videos, and their mixed multimodal unstructured data.

e) It shall support data import from external databases in the form of full volume, incremental, streaming, etc., such as relational databases.

f) It shall support annotation tools that integrate or provide training and validation data.

g) It can support automated or semi-automated knowledge acquisition, such as machine learning, dictionaries, and rules.

h) It can support manual input and editing of knowledge.

i) It shall support automatic, semi-automatic, and manual proofreading and verification.

j) It shall support knowledge acquisition in open-source ontologies or knowledge graphs.

k) It should support event extraction.

NOTE—Not limited to textual, multimodal, case (reasoning) graphs, etc.[11]

5.3.2 Performance requirements

The performance metrics of knowledge acquisition related modules include, but are not limited to, the following:

a) Precision, which is to measure the ratio of acquired knowledge that is consistent with real-world knowledge and has no gaps in the process of acquiring knowledge. Precision is calculated in Equation (9).

$$\text{Precision}_A = \frac{TP_A}{TP_A + FP_A} \tag{9}$$

where

TP_A is the true positive, which is the number of entities, relationships, or attributes that have been identified and match the truth

FP_A is the false positive, which is the number of entities, relationships, or attributes that are recognized but do not match the truth

b) Recall, which is to measure the degree that acquired knowledge covers correct knowledge. Recall is calculated in Equation (10).

$$\text{Recall}_A = \frac{TP_A}{TP_A + FN_A} \tag{10}$$

where

FN_A is the false negative, which is the number of entities, relationships, or attributes marked as real but not recognized

c) F1-Measure, which is to comprehensively measure the precision and completeness of knowledge acquisition results. F1-Measure is calculated in Equation (11).

$$\text{F1-score}_A = 2 \times \frac{\text{Precision}_A \times \text{Recall}_A}{\text{Precision}_A + \text{Recall}_A} \tag{11}$$

[11]Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

d) Comprehensiveness of knowledge acquisition

   1) Whether the knowledge covers all entities within the field

   2) Whether the knowledge covers all relationships between entities

   3) Whether the knowledge covers all attributes of an entity

e) Relationship constraint verification, such as:

   1) Whether it is a 1-to-1 relationship (according to the ontology model relationship definition)

   2) Whether it is a 1-to-n relationship (according to the ontology model relationship definition)

   3) Whether it is a many to many relationship (according to the ontology model relationship definition)

f) Attribute constraint verification, such as:

   1) Whether the attribute has uniqueness (according to the ontology model attribute definition)

   2) Whether the attribute is required to be non-empty (according to the ontology model attribute definition)

      NOTE—For the overall ability evaluation of knowledge acquisition, entity attribute/relationship entity combination can be considered as a set of evaluation data to evaluate precision, recall, and F1-Measure.

5.4 Knowledge storage

5.4.1 Functional requirements

The functional requirements of knowledge storage related modules include, but are not limited to, the following:

a) It shall support the storage and management of multi-version knowledge graphs.

b) It shall support the creation and storage of indexes.

c) It shall support storing log records.

d) It shall support adaptation of different storage engines.

e) It shall support storage encryption.

f) It shall support knowledge graphs disaster recovery.

g) It shall support transaction consistency in knowledge storage process, such as resolving data conflicts in concurrent situations.

h) It shall support distributed storage, including graph partitioning, distributed transactions, online expansion, load balancing, etc.

i) It shall support multiple attribute data types, such as numerical, string, boolean, and other commonly used data types.

j) It shall support multimodal data types.

k) It shall support at least one graph model, such as RDF graph, attribute graph, hypergraph, etc.

l) It shall support multiple table models, such as triplet tables, type tables, level tables, DB2RDF, etc.

m) It shall support batch export of data, incremental update (including adding, deleting, modifying attributes, etc.), and incremental export.

n) It shall support automatic updating of schemas or data triggered by knowledge acquisition, knowledge fusion, and knowledge modeling.

o) It shall support the update and modification of knowledge, such as:

1) Regular updates

2) Knowledge correction

3) Knowledge modification

p) It shall support updating stored knowledge content without stopping the query service.

q) It shall support real-time knowledge graph updates.

r) It shall support knowledge query when updating knowledge content.

s) It shall support basic queries (such as detailed queries of points and edges), fuzzy queries, higher-order queries (such as paths and subgraphs), addition, deletion, modification, display, and other operations.

t) It can support multiple query languages.

u) It shall support API and application side interaction, such as ODBC, JDBC, RPC, RESTful, graph query language, constraint query language, etc.

v) It shall have functions such as graph traversal, shortest path query, minimum spanning tree search, rank value calculation, community discovery, etc.

w) It shall support display through a graphical interface.

5.4.2 Performance requirements

The testing metrics for knowledge storage related modules include, but are not limited to, the following:

a) Reading performance related metrics:

1) Data format types: commonly used data types such as numerical, string, Boolean, etc.

2) The information density of stored knowledge, which is to measure the density of knowledge units in the storage space. The information density of stored knowledge is calculated in Equation (12).

$$V_R = \frac{\sum k_u}{M} \tag{12}$$

where

$\sum k_u$ is the total number of storage knowledge units (nodes and relationships)
M is the storage space occupied

3) K-hop neighbor query response time, which is to measure the query time of the system for associated objects in different levels of expansion and connection, such as 1-hop, 2-hop, and K-hop. K-hop neighbor query response time is calculated in Equation (13) and Equation (14).

$$X_i - \mathrm{Rec}_i - \mathrm{Send}_i (i \sim n) \tag{13}$$

$$Y_i = (X_i - X_T)(i \sim n, X_i > X_T) \tag{14}$$

where

X_i is the response time of the i-th K-hop query
Rec_i is the time when the i-th K-hop query received the response result

Send_i is the time when the i-th K hop query request is issued
Y_i is the timeout of the i-th K hop query
X_T is the timeout threshold of K hop query
n is the total number of the tests

The constraint conditions are described in the following table:

The total amount of entities and relationships	Number of K hops	Query response time range
Up to one million	2 hops	Less than 0.5 s
Up to 10 million	2 hops	Less than 1 s
Up to 100 million	3 hops	Less than 2 s
More than 100 million	3 hops	Less than 10 s

b) The maximum number of occurrences for K-hop neighbor queries, which is to measure the maximum number of K-hop query requests that the system can withstand at the same time. Exceeding this number of requests will lead to a serious decline in system efficiency and may lead to system failure. The maximum number of occurrences for K-hop neighbor queries is calculated in Equation (15).

$$X_{\max} = MAX(N_r[i]) \tag{15}$$

where

$X_{\max}$ is the maximum number of concurrent requests for K hop queries
$N_r[i]$ is the number of concurrent requests that need to be tested in ascending order. It is required that under this test, the response time can still be accepted and has not increased sharply, or the throughput rate has not reached the extreme value to become a bottleneck

The constraint is feedback results within 30 s.

c) Millisecond level data query response rate, which is to measure the number of response times below the millisecond level during a query operation. Millisecond level data query response rate is calculated in Equation (16).

$$X_R = \frac{A_R}{B_R} \tag{16}$$

where

A_R is the number of times the query response times which are below the millisecond level
B_R is the total number of times which the query in graph

1) Update time, which is to measure the time taken to add, delete, or modify entities or relationships in the system. Update time is calculated in Equation (17).

$$T_i = \text{Rec}_i - \text{Send}_i(i \sim n) \tag{17}$$

where

T_i is the total time required at the i-th update operation
Rec_i is the time when the completion response is received at the i-th update operation
Send_i is the time when the request is issued at the i-th update operation

d) Write performance related indicators

1) Throughput rate, which is to measures the number of requests processed by the system per unit of time under a specific number of concurrency. Throughput rate is calculated in Equation (18).

$$RPS = \frac{CR_g}{T_g} \tag{18}$$

where

CR_g is the total number of requests
T_g is the total completion times for processing these requests

The constraint is that the first degree neighbor throughput rate is not less than 10 under the concurrent number of 20.

2) Data loading/importing time, which is to measure the time (in seconds/milliseconds) that the system completes loading the benchmark data set and guarantees that the data loading results are correct. Data loading/importing time is calculated in Equation (19).

$$X_i = \mathrm{Rec}_i - \mathrm{Send}_i (i \sim n) \tag{19}$$

where

X_i is the response time for the i-th load
Rec_i is the time when the response result is received for the i-th load
Send_i is the time when the i-th load data request is issued
n is the total number of the tests

NOTE—The constraint is that the loading rate is greater than or equal to 5 GB/h.

3) QPS (single instruction), which is to measure the number of times a system can respond to a single instruction query per second. The constraint is that QPS (single instruction) is greater than or equal to 100.

4) QPS (multiple instructions), which is to measure the number of times a system can respond to multiple instruction queries per second. The constraint is that QPS (two instructions) is greater than or equal to 50.

5) Metrics of transaction consistency (data conflicts in concurrent situations), which is to measure the ability of the system to guarantee data consistency in concurrent situations. The following indicators can be used:

i) Transaction conflict rate (TCR), which is to measure the frequency of conflicts caused by concurrency during the execution of a transaction. TCR is calculated in Equation (20).

$$TCR = \frac{N_{conflict}}{N_{total}} \tag{20}$$

where

$N_{conflict}$ is the number of transactions that conflict during concurrent operations
N_{total} is the total number of transactions

ii) Transaction Rollback Rate (TRR), which is to measure the ratio of transactions that must be rolled back due to conflicts. TRR is expressed in Equation (21).

$$TRR = \frac{N_{rollback}}{N_{total}} \tag{21}$$

where

$N_{rollback}$ is the number of transactions that must be rolled back due to conflicts
N_{total} is the total number of transactions

5.5 Knowledge fusion

5.5.1 Functional requirements

The functional requirements of knowledge fusion related modules include, but are not limited to, the following:

a) It shall support entity linking, entity disambiguation, and attribute disambiguation of raw extracted data.

NOTE—Resolve conflict issues based on alignment, such as matching confusion.

b) It shall support instance level alignment (entities, relationships, and their attribute values), that is, establishing associations between different knowledge bases and data sources.

c) It shall support entity type alignment, attribute name alignment, and relationship type alignment fusion to guarantee consistency between different levels of knowledge graphs.

d) It shall support the fusion of multiple ontologies.

e) It shall support the fusion of rules, functions, and other logics, providing a shared inference mechanism between different knowledge bases and data sources.

f) It shall support real-time fusion of new knowledge.

g) It shall support alignment and fusion of knowledge graphs across languages (such as Chinese and English).

h) It shall support consistency/conflict detection of entities or relationships to be fused.

i) It shall support the inheritance of attributes in entities and relationships, and retain the original attributes and relationships for knowledge fused from different knowledge bases and data sources.

j) It should support the fusion of heterogeneous multi-source entities, and integrate the same entities that occur in different knowledge graphs based on interpretable and traceable rules.

k) It should support knowledge fusion based on machine learning technologies such as deep learning or embedded learning.

5.5.2 Performance requirements

The performance metrics of knowledge fusion related modules include, but are not limited to, the following:

a) Precision, which is to measure the ratio of correct knowledge to the total scale of fused knowledge in the result of knowledge fusion. Precision is calculated in Equation (22).

$$\text{Precision}_F = \frac{TP_F}{TP_F + FP_F} \tag{22}$$

where

TP_F is the true positive, which is the number of entities, relationships, or attributes in the fusion result that match the truth

FP_F is the false positive, which is the number of entities, relationships, or attributes in the fusion result that have been fused but do not match the truth

b) Recall, which is to measure the degree to which the fused knowledge covers the correct knowledge in the result of knowledge fusion. Recall is calculated in Equation (23).

$$\text{Recall}_F = \frac{TP_F}{TP_F + FN_F} \tag{23}$$

where

FN_F is the false negative, which is the number of entities, relationships, or attributes to be fused but not fused in the fusion result

c) Top k hit rate, which is to measure the hit rate of the top k target knowledge in the knowledge fusion result. Top k hit rate is calculated in Equation (24).

$$Hit@k_F = \frac{NH@k_F}{GT_F} \tag{24}$$

where

$NH@k_F$ is the number of entities, relationships, or attributes in the knowledge fusion result that match the first k target knowledge, which is given by the actual knowledge fusion task
GT_F is the total number of entities, relationships, or attributes related to the target knowledge

d) F1 measure, which is to comprehensively measure the accuracy and completeness of knowledge fusion results. F1 measure is expressed in Equation (25).

$$\text{F1}-\text{score}_F = 2 \times \frac{\text{Precision}_F \times \text{Recall}_F}{} \tag{25}$$

e) Fusion efficiency, which is to comprehensively measure the scale of knowledge fusion per unit time. Fusion efficiency is expressed in Equation (26).

$$\text{efficiency}_F = \frac{K_F}{T_F} \tag{26}$$

where

K_F is the total amount of fused entities, relationships, or attributes
T_F is the time-consuming of fusion process

5.5.3 Functional requirements

Functional requirements for knowledge computing related modules include, but are not limited to, the following:

a) Support basic graph calculation functions, such as multi-graph comparison, single graph calculation, connected subgraph calculation, etc.
   1) Shall support graph calculation functions such as entity node ranking, degree centrality calculation, closeness centrality calculation, betweenness centrality and other graph calculation functions
   2) Shall support graph computing functions such as community discovery and minimum spanning tree
   3) Shall support graph calculation functions, such as connected graphs
   4) Should support graph similarity calculation functions, such as intersection/union
   5) Shall support graph query and retrieval functions

b) Support node computing functions, include the following:
   1) Shall support knowledge statistics, including single/multi-node degree (including in-degree, out-degree), given node statistics, etc.
   2) Shall support statistical clustering coefficient distribution (correlation analysis), used to measure the trend of node aggregation

3) Should support node anomaly detection

c) Support path calculation functions, include the following

1) Shall support the set of knowledge graphs paths giving start nodes and target nodes

2) Shall support calculating the shortest path between two nodes based on the specified relationship type

3) Should support the calculation of path feature weights to distinguish the closeness of interaction between different paths

d) Shall support knowledge completion, such as graph relationships completion, node relationships completion, logic links reasoning, etc.

e) Should support the integration or compatibility of symbolic computing, neural networks, etc.

f) Should support distributed computing

g) Should support reasoning and calculation of events

5.5.4 Performance requirements

Performance test indicators of knowledge computing related modules shall include, but are not limited to, the following:

a) Response time, which is to measure the time it takes for a knowledge computing task from start to end. Response time of knowledge computing is expressed in Equation (27).

$$t_c = t_{end} - t_{start} \tag{27}$$

where

t_{start} is start time of the knowledge computing task
t_{end} is end time of the knowledge computing task

b) Precision, which is to measure the degree to which the knowledge computing results are consistent with the preset target values. Precision of knowledge computing is expressed in Equation (28).

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \tag{28}$$

where

TP_c is true positive: the number of knowledge computing results that are consistent with the preset target values
FP_c is false positive: the number of knowledge computing results that are inconsistent with the preset target values

c) Logical language richness measures the number of supported logical languages

d) Rule engine richness measures the number of supported rule engines

e) Software and hardware resource consumption measures the maximum software and hardware resources consumed during the execution of computing tasks

f) Compatibility measures whether the knowledge computing algorithm can be compatible or inherited with existing algorithm models, basic software, and hardware resources, etc.

5.6 Knowledge provenance

5.6.1 Functional requirements

Performance test metrices of knowledge provenance related modules include, but are not limited to, the following:

a) Should support recording of original data processing processes

b) Shall record the source of original data

c) Shall support the monitoring and management of data production processes

5.6.2 Performance requirements

Performance metrics of knowledge provenance related modules shall include, but are not limited to, the following:

a) Precision of knowledge provenance, which is to measure the degree of consistency between the knowledge provenance results and the real target values. Precision of knowledge provenance is calculated in Equation (29).

$$\text{Precision}_T = \frac{TP_T}{TP_T + FP_T} \tag{29}$$

where

TP_T is true positive: the number of knowledge provenance results that are consistent with the true target values

FP_T is false positive: the number of knowledge provenance results that are inconsistent with the true target values

b) Completeness of knowledge provenance, which is to measure the coverage level of the knowledge provenance results on all real target knowledge. Completeness of knowledge provenance is calculated in Equation (30).

$$\text{Completeness}_T = \frac{TP_T}{GT_T} \tag{30}$$

where

GT_T is the total amount of true target knowledge

c) Confidence of knowledge provenance measures the confidence level of the knowledge provenance results, that is, the probability that the knowledge provenance results are consistent with the true target values. Confidence of knowledge provenance is calculated in Equation (31).

$$\text{Confidence}_T = \frac{M_T}{N_T} \tag{31}$$

where

M_T is the number of times the knowledge provenance results are consistent with the real target values in several knowledge tracing tests

N_T is the number of repetitions of the knowledge provenance test

5.7 Knowledge evolution

The functional requirements for a knowledge evolution module include, but are not limited to, the following:

a) Shall support the version management of knowledge graphs

b) Shall support the knowledge version setting of a current external service

c) Shall support the cancellation or rollback of a historical knowledge version

d) Shall support the revocation of a historical knowledge version operation

e) Should support the resolution of conflicts among multi-version knowledge graphs

f) Should keep a knowledge evolution log

g) Should record operator/editor information in the evolution process

5.8 Knowledge visualization

5.8.1 Functional requirements

The functional requirements for a knowledge visualization module include, but are not limited to, the following:

a) Graphical representation shall be supported to determine the representation forms of entities, relationships, and attributes in the graphs.

b) Shall support browser-based data representation forms.

c) Shall support customized entity and relationship representation forms, such as node shape, head portrait, size, etc.

d) Should support paging display.

e) Should support multi-dimensional display of the same data from the same source.

f) Shall support dragging of nodes or edges.

g) Shall support the hierarchy or layout setting of knowledge.

h) Shall support interaction with users through graphical interfaces.

i) Should support the creation of ontology models by dragging or clicking.

j) Should support interactive interface operations, such as dragging, clicking, adding, deleting, modifying, and querying, etc.

5.8.2 Performance requirements

The performance metrics of a knowledge visualization module shall include, but are not limited to, the following:

a) Rendering speed X_p is calculated in Equation (32).

$$X_p = \frac{N_p}{T_p} \tag{32}$$

where

N_p is the number of graphical nodes or relationships within T_p, the time taken for rendering

When the number of concurrent transactions is 10, the constraints are described in the following table:

Total number of rendered entities and relationships	Average rendering time
No more than 50	Less than 5 s
No more than 100	Less than 10 s
No more than 1000	Less than 20 s
More than 1000	Less than 30 s

b) The maximum number of nodes that should be represented (within a specified period of time, such as 10 s)

c) Operation fluency: the module should respond to the user's clicking on a single entity or relationship within 1 s

d) Response speed: the speed of response to one-time loading, editing, hierarchical display and other operations

e) Hardware resource consumption: the maximum memory and computing resources consumed during the execution of computing tasks

5.9 Knowledge exchange

The functional requirements for a knowledge exchange module include, but are not limited to, the following:

a) Shall support the import of normative knowledge graph files

b) Shall support the export of knowledge graphs or selected subgraphs

c) Shall support the encryption and decryption of knowledge graph files

d) Should support network-based knowledge content transmission and interaction

e) Shall support the encryption and decryption of knowledge during the network-based transmission and interaction of knowledge content

6. Requirements for modules related to knowledge graph applications

6.1 Responsiveness

6.1.1 Knowledge graph scale

Knowledge graph scale-related evaluation metrics shall include but not be limited to the following:

a) Knowledge graph volume measures the number of entities and relationships contained in the knowledge graph. The calculation formula is as follows:

$$V = V_e + V_r \tag{33}$$

where

V_e is the number of entities in the knowledge graph
V_r is the number of relations in the knowledge graph

NOTE—For attribute graphs, measure the attribute values. For time series, geography, etc., perform separate statistics.

b) Knowledge graph complexity measures the number of entity types and relationship types contained in the knowledge graph. The calculation formula is as follows:

$$F = F_e + F_r + F_a \tag{34}$$

where

F is the number of entity types in the knowledge graph
F_r is the number of relationship types in the knowledge graph
F_a attribute items in the knowledge graph

NOTE—For different fields, the complexity can be further subdivided and counted.

6.1.2 Knowledge graph quality

Knowledge graph quality-related evaluation metrics shall include, but not be limited to the following:

a) Knowledge coverage: Refers to the evaluation of the proportion of knowledge contained in a knowledge graph within a specific domain or application scenario to all relevant knowledge within this scope. Knowledge coverage is typically used to measure the completeness and validity of a knowledge graph for use in various application scenarios. The calculation formula is as follows:

$$CD = \frac{N_{ent}}{\alpha D_{aut} + (1 - \alpha) D_{pra}} \times 100\% \tag{35}$$

where

D_{aut} is the number of entities with authoritative evidence in the domain, such as guidelines, standards, policies, etc
D_{pra} is the number of entities with practical evidence, such as localization experience, special processes, or anomalous processes, etc
α is a weight parameter that moderates the global impact of authoritative and practical evidence
N_{ent} denotes the number of entities appearing with authoritative and practical evidence in the knowledge graph

b) Accuracy of intellectual reasoning: measures the reasoning ability in the knowledge graph application scenarios. It can be evaluated by calculating the accuracy and recall of reasoning results based on evaluation test data sets.

1) Accuracy rate: It measures the proportion of correct conclusions produced by knowledge-based reasoning over all reasoned conclusions. The calculation formula is as follows:

$$\text{Precision}_R = \frac{TP_R}{TP_R + FP_R} \tag{36}$$

where

TP_R T is true positives; the number of correct reasoning outcomes
FP_R is false positives; the number of incorrect reasoning outcomes

2) Recall rate: measures the proportion of correct conclusions produced by knowledge-based reasoning over all correct conclusions that shall be inferred. The formula shall satisfy:

$$\text{Recall}_F = \frac{TP_F}{TP_F + FN_F} \tag{37}$$

where

FN_F is false negatives; the amount of correct knowledge failed to be inferred
TP_F is true positives; the amount of correct knowledge succeeds to be inferred

c) Knowledge trustworthiness: knowledge trustworthiness is a measure of the reliability and accuracy of knowledge entries in a knowledge graph. Knowledge credibility can be measured in several ways:

1) Data sources: It evaluates the authority of knowledge sources. Knowledge from authoritative, specialized, and reliable data sources has a higher degree of credibility.

2) Number of citations: It evaluates how often the knowledge is cited in other literature, data, or knowledge graphs. Knowledge that is widely cited may have higher credibility.

3) Expert assessment: Experts are asked to assess and validate the knowledge to determine its credibility. The results of the expert assessment can be used as a basis for measuring credibility.

4) Knowledge consistency: It checks whether relevant knowledge in the knowledge graph is consistent with each other. Knowledge with higher consistency is likely to have higher credibility.

d) Knowledge timeliness: A measure of the novelty and timeliness of knowledge entries in a knowledge graph. Knowledge timeliness can be measured in several ways:

1) Frequency of updates: It evaluates how often the knowledge graph is updated. Knowledge graphs that are updated at a high frequency are likely to be highly current.

2) Data generation time: It evaluates the generation time of the knowledge in the knowledge graph. Knowledge that is closer to the current time is more effective.

3) Speed of data change: It evaluates the rate of data change in a given domain or application scenario. Domains with faster rates of data change require higher timeliness of knowledge.

4) Timeliness requirements: The importance of knowledge timeliness is assessed based on the specific needs of the application scenario. For scenarios with high timeliness requirements, the knowledge graph needs to maintain high knowledge timeliness.

5) Real-time: It evaluates the extent to which the existing knowledge contained in the knowledge graph in a knowledge graph-based assisted decision-making system is consistent with the latest knowledge of the real-world state. The formula is shown as follows:

$$KT = \left(1 - \frac{n}{N_r + N_a}\right) \times 100\% \tag{38}$$

where

n is the number of relationships and attributes in the knowledge graph that do not correspond to real-world states
N_r is the total number of relationships in the knowledge graph
N_a is the total number of attributes in the knowledge graph

6.1.3 Response efficiency

Response efficiency measures the number of requests processed within a specified time. The calculation formula is as follows:

$$E = \frac{A_R}{T} \tag{39}$$

where

A_R is the total number of requests processed
T is the time used to process the request

6.1.4 Response accuracy rate

Response accuracy rate-related evaluation metrics shall include but not be limited to the following:

a) Response accuracy rate: measures the proportion of results that meet the request (such as: fuzzy query, semantic search, path reasoning, etc.) to the total number of returned results. The calculation formula is as follows:

$$\text{Precision}_R = \frac{TP_R}{TP_R + FP_R} \tag{40}$$

where

TP_R is true positives; the number of responses that were identified and matched true
FP_R is false positives; the number of responses that were identified but did not match the real one

b) Response recall rate: It measures the degree of correct results in the results of request feedback. the calculation formula shall satisfy:

$$\text{Recall}_F = \frac{TP_F}{Tp_F + FN_F} \tag{41}$$

where

FN_F is false negatives; the number of correct results that were not identified in the feedback results

6.1.5 Real-time responsiveness

Real-time responsiveness-related evaluation metrics shall include but not be limited to the following:

a) Real-time responsiveness: measures the response ratio of the knowledge graph construction and application system to requests sent by users within a limited time. The calculation formula is as follows:

$$X_A = \frac{A_R}{A} \tag{42}$$

where

A_R is the number of times a request is responded within a limited time
A isthe total number of requests made

b) When querying and editing, the real-time response time shall not exceed 400ms.

c) During visual interaction:

1) Query function: the user enters a keyword or a question, and the system returns relevant entities, relationships, attributes, and other information. The recommended response time is within 0.5 s.

2) Analysis function: the user inputs a topic or a range, and the system returns relevant statistics, clustering, classification, and other information. The recommended response time is within 5 s.

3) Inference function: the user inputs a hypothesis or a goal, and the system returns relevant evidence, paths, conclusions, and other information. The recommended response time is within 10 s.

6.2 Friendliness

6.2.1 Usability

Usability refers to the features that knowledge graph applications can meet the needs and expectations of users when designed and implemented, and provide a simple, efficient, and pleasant user experience.

Requirements related to usability include, but are not limited to the following:

a) Effectiveness: means that knowledge graph applications can help users complete specific tasks and goals and achieve expected results and quality.
   1) It shall have a human-computer interaction interface for all aspects of the entire life cycle of the knowledge graph.
   2) It shall support map data to display the relationship between entities or concepts in dimensions such as time, space, or topology.
   3) It shall support the import, export, addition, deletion and modification of data through a visual interface.
   4) It shall support fuzzy search/exact match search of entities and attributes.
   5) It shall support a variety of interaction methods, such as: interaction based on natural language, video, pictures, etc. during terminal use.
   6) It can be adapted to different groups of people/roles (such as users with different permissions and administrators).

b) Efficiency: refers to whether the time, energy, resources, etc. spent by users when using knowledge graph applications are reasonable and economical.

c) Satisfaction: refers to the subjective satisfaction and pleasure that users feel when using knowledge graph applications.

d) Learnability: refers to whether the time, difficulty, help, etc., required for users to use or learn a knowledge graph application for the first time are simple and convenient.

e) Memorability: refers to whether the recall, review, and re-learning required by users when they use the knowledge graph application again after a period of time are easy and fast.

6.2.2 Inheritability

Inheritability refers to the characteristic that knowledge graph applications shall support the reuse and expansion of knowledge and improve the value and efficiency of knowledge when designing and implementing it.

Inheritability-related requirements include, but are not limited to the following:

a) Knowledge reuse: means that knowledge graph applications can use existing knowledge to solve new problems or meet new needs, reducing the duplicate construction and waste of knowledge.
   1) It shall have functions such as import, export, and fusion of top-level schemas.
   2) It shall support import and export of knowledge data.
   3) It shall support the sharing of knowledge systems/example collections, and can be based on visual interfaces, interfaces, and documents.
   4) It shall support the reference of the knowledge system/example collections, and can be based on visual interfaces, interfaces, and documents.

b) Knowledge expansion: means that knowledge graph applications can add or modify knowledge based on new questions or needs, improving the completeness and accuracy of knowledge.

   1) It shall support the functions of adding new entity types, relationship types, attribute names, etc.

   2) It shall support functions other than export and fusion.

   3) It shall support editing, modification and other operations based on references.

6.2.3 Maintainability

Maintainability refers to the characteristic that knowledge graph applications, when designed and implemented, can support knowledge updates and corrections, as well as system status monitoring, updates, and corrections, improving the quality and reliability of the system.

Requirements related to maintainability include, but are not limited to the following:

a) It shall have functions such as monitoring, early warning or alarm, and provide feedback the operating status of the system.

b) It shall support operation logs and system log records.

c) It shall set necessary check points.

d) It shall provide supporting maintenance documents, data description documents, and use documents of functional modules.

e) It shall keep the system functionality clear.

f) It shall support the management of data extraction, data classification, data cleaning and other links.

g) It should support the management of knowledge application, integration, and storage.

6.3 Reliability

6.3.1 Availability

Availability refers to the probability that the system can work normally within a given time. It reflects the system's ability to quickly resume normal operation after a failure.

NOTE—"Unavailable" refers to the system's inability to provide services, such as queries or editing, when it is not functioning. On the other hand, "available" means that the application layer recovers from a crash, resumes providing services, and eventually returns to a working state.

Availability-related test metrics shall include but are not limited to average failure time within a given period (e.g., 30 d). The indicators include the following:

a) A average failure time: Measures the duration the system is in an unavailable state when a failure occurs, with the calculation formula as follows:

$$X_R = \frac{T_F}{N_F} \tag{43}$$

where

T_F is the total duration of unavailable state
N_F is the number of failures

NOTE—Types of failures include single machine mode (which includes network disconnection, power outage, service restart, database restart, etc.) and distributed mode (which includes service unavailability, etc.).

b) Service level agreement (SLA): SLA is a contract between the service provider and the customer, defining the type and standard of the service to be provided. It is not less than 99.9% availability. The subjects of availability can include cloud services, software tools (private deployment), etc.

c) Average failure repair time: Recovery time for single users and multiple users shall not exceed 10 min.

d) Number of access control permissions.

6.3.2 Maturity

Maturity refers to the extent to which a system or product can achieve the expected level and objectives of reliability during its design and implementation process.

Maturity-related test indicators include but are not limited to the following:

a) Fault elimination rate: Measures the circumstances under which system failures can be eliminated and repaired, with the calculation formula as follows:

$$X_C = \frac{N_C}{N_F} \tag{44}$$

where

N_C is the number of corrected faults
N_F is the number of faults that occurred during testing

b) Mean time between failures (MTBF): Measures the duration between failures of the system operating under specified working conditions and environments within a certain operational time range, with the calculation formula as follows:

$$T_F = \frac{T_R}{N_F} \tag{45}$$

where

T_R is the system operating time
N_F is the number of system failures

c) Failure rate: Average number of failures within a specified period, with the calculation formula as follows:

$$N_{FA} = \frac{N_F}{T_R} \tag{46}$$

where

N_F is the number of system failures
T_R is the system operating time

6.3.3 Fault tolerance

Fault tolerance refers to the ability of a system or product to continue providing services when encountering faults or anomalies. It reflects the system's ability to resume normal operation promptly after a fault occurs.

Fault tolerance-related requirements include but are not limited to the following:

a) Measures shall be set up to guard against incorrect operations, such as necessary reminders, confirmations, and cancellations, to avoid system faults or data loss due to user errors.

b) Critical and severe failures, such as system crashes, data corruption, and loss of functionality, shall be avoided to guarantee the normal operation of the system and user satisfaction.

c) The system shall guarantee that failures do not cause downtimes, using technologies such as redundancy, backup, and switching, to achieve high availability and continuity.

d) The system shall implement emergency disaster recovery, using technologies such as recovery, restart, migration, etc., to achieve rapid system recovery and security.

6.3.4 Recoverability

Recoverability refers to the extent to which a product or system can restore directly affected data and rebuild the desired system state when interruptions or failures occur. It reflects the system's ability to promptly resume normal operation after a fault.

Recoverability-related requirements include but are not limited to the following:

a) Support for key data backup shall be provided, such as export, snapshots, etc.

b) Rollback of key data shall be supported.

c) Support for recovery from deviations caused by errors during data operations, data source readings, etc., shall be available.

d) Errors during data use shall be recoverable.

e) Recovery during data collection, such as data redundancy, dirty data, etc., shall be supported.

f) Recovery during data transmission shall be supported.

g) Support for atomicity (indivisibility) of key operations shall be available.

h) Support for transactionality of business operations: when the Knowledge Graph application performs business operations, such as batch modifications, batch deletions, or power outage operations, it is required to guarantee data consistency, isolation, persistence, and atomicity. Transactional operations typically perform data backup and recovery based on fixed time points or event triggers to guarantee system state correctness.

i) Configurable rapid recovery measures or fault alert measures shall be available during system failures.

j) Automatic system repair shall be supported under exceptional circumstances or when needed.

k) The main quantifiable indicators related to recoverability include:

    1) Version updates and iterations within 24 h

    2) Recovery time

    3) Response time, etc.

6.4 Portability

Portability refers to the ability of a system or product to move from one environment to another. It reflects the adaptability and flexibility of the system. Portability-related requirements include but are not limited to the following:

a) Support for commonly used hardware environments shall be provided, such as different processor architectures.

b) Support for commonly used operating systems shall be provided.

c) Support for commonly used databases shall be provided, such as graph databases, relational databases, non-relational databases, etc., to accommodate different data sources and formats.

d) Support for commonly used programming languages and frameworks shall be provided.

e) Support for commonly used document formats and data formats shall be provided.

f) Support for commonly used network protocols and interfaces shall be provided.

g) Data import and export shall be supported during the porting process, such as providing data conversion tools or interfaces, to guarantee data consistency and completeness in different environments.

h) Functional testing and compatibility testing shall be supported after porting, such as providing test cases or tools, to guarantee the correctness and stability of the system in different environments.

i) Portability can be measured by quantifiable indicators such as average installation time and installation success rate. The calculation formula for the installation success rate is as follows:

$$X_I = \frac{N_{I\_S}}{N_I} \tag{47}$$

where

N_{I_S} is the number of successful installations
N_I is the total number of installations

The constraint condition is that all data migration is successful.

6.5 Security

Security-related requirements include but are not limited to the following:

a) Integrity

1) The system shall support recording changes to knowledge such as entities, relationships, attributes, etc., generated from client data during the use of the knowledge graph system.

2) The system shall not support unauthorized implantation, tampering, replacement, transfer, and forgery of knowledge such as entities, relationships, attributes, instances, etc., generated from client data.

3) The system shall support setting permissions for different roles.

4) The system shall support recording changes to the functional points specified by the client in the knowledge graph system.

5) Functional integrity protection shall be supported to avoid unauthorized tampering, replacement, and reduction.

6) The system can record the complete state of knowledge during service updates and increments in the knowledge graph system.

b) Controllability

1) It is advisable to support defenses against adversarial sample attacks and to monitor and avoid data poisoning (related to machine learning model processes).

2) Support for monitoring and avoiding anomalies in the knowledge graph caused by data replacement and reduction shall be provided.

3) Support for topology security, system security, and management security shall be provided.

4) Interface authentication shall be supported to avoid unauthorized access and invocation.

c) Confidentiality

1) Support for encryption algorithms and protocols during the knowledge transfer process shall be available.

2) Support for encrypted transmission during data transfer shall be available.

3) Encryption of data shall be supported.

d) Privacy

1) Privacy protection measures shall be in place to avoid information leakage.

2) Support for permission management, such as hiding the ontology, shall be provided.

e) Auditability

1) Support for the auditability of operations and content in the knowledge graph shall be available.

2) It shall comply with ethical, intellectual property, and legal requirements.

7. Testing environment

The testing environment for the construction and application system of a knowledge graph shall meet the following requirements and shall have deployed test scenarios that are compatible with the system under test. The testing environment mainly includes the following:

a) Testing environment

1) Environmental temperature of the testing facility: 15 °C to 35 °C.

2) Relative humidity of the testing facility: 25% to 75%.

3) Operating systems: Operating systems based on x86 or ARM architectures.

4) Client web browsers (optional).

b) Server environment

1) Total CPU cores of the test server: Greater than or equal to 32 cores (CPU clock speed-percentile).

2) Total memory of the test server: Greater than or equal to 256 GB (memory read/write frequency-percentile).

3) Total hard disk space of the test server: Greater than or equal to 1 TB (generally, high reliability and high read/write speed disks are chosen, such as SSDs or enterprise-grade mechanical hard drives with a read/write speed of no less than 200 MB/s).

4) Intelligent acceleration card (optional): Computational performance, such as GPU, NPU, etc.

NOTE—For distributed or standalone systems, the scale can range from tens of millions and below; for systems in the billions and above, requirements are increased accordingly.

c) Types of server environments.

1) Standalone: Direct connection between the client and a single server (network bandwidth between servers: 1 Gbps or higher).

2) Distributed: Comprising multiple servers within the same local network.

3) Cloud environment: Testing conducted on a cloud platform, with no less than the total CPU, total memory, and total hard disk resources.

d) Testing tools: Appropriate testing tools are required to assess the system's performance and reliability.

e) Test cases: The test cases about data input, knowledge modeling, knowledge acquisition, knowledge fusion, knowledge storage and retrieval, inference and analysis, and knowledge graph display, etc. Can be prepared according to the testing methods given in Clause 8 and Clause 9. Annex B gives typical test cases.

NOTE—Annex A provides the testing process of the knowledge graph construction and application system. The performance testing methods for knowledge graph construction and application related modules show in Clause 8 and Clause 9. The function requirements in Clause 6 and Clause 7 can be judged whether the system has the corresponding functions, which is not described further.

8. Performance testing method for knowledge graph construction related modules

8.1 Performance testing method for knowledge modeling

8.1.1 Inputs semantic clarity of ontology models

The testing steps for this feature include the following:

a) Open the relevant modules for knowledge modeling.

b) Randomly select B_{O_c} entity types, relationships, or attributes from the ontology model constructed.

c) Filter out the corresponding raw data.

d) Determine the number of clearly defined entity types, relationships, or attributes A_{O_c} in the extracted results.

e) Calculate the ratio of A_{O_c} to B_{O_c} in the extracted results.

NOTE—If conducting a cross-comparison process, it is recommended to use benchmark data sets. If testing is performed using data sets provided by the tested party, it is necessary to declare the source of the test set, its scale, and the distribution of samples in each category.

8.1.2 Ontology model completeness

There are two types of methods for testing the completeness of an ontology model:

a) Completeness testing based on the ontology model: This method involves sampling knowledge from the ontology model and comparing it with the scope of the ontology knowledge system and the range of knowledge sources. It verifies whether the designed ontology model is complete. The specific testing steps include:

1) Open the relevant knowledge modeling modules.

2) Determine the sampling method and proportion of knowledge in the ontology model based on constraints such as the scope of the ontology knowledge system and the range of knowledge sources, e.g., uniform sampling, normal distribution sampling, random sampling, etc.

3) Extract $B_{O_{D1}}$ entities, relationships, or attributes from the constructed ontology model.

4) Evaluate the extracted results manually or by using software-assisted evaluation to determine the number of correct entity types, relationships, or attributes, denoted as $A_{O_{D1}}$.

5) Calculate the ratio of $A_{O_{D1}}$ to $B_{O_{D1}}$ in the extracted results.

NOTE—In some cases, industry experts may also participate in the evaluation.

b) Completeness testing based on entities and relationships: This method involves sampling entities and relationships and reverse-verifying whether the designed ontology model is complete. The specific testing steps include:

1) Open the relevant knowledge modeling modules.

2) Determine the sampling method and proportion of entities, relationships, or attributes in the knowledge graph extracted according to the designed ontology model, based on constraints such as the scope of the ontology knowledge system and the range of knowledge sources, e.g., uniform sampling, normal distribution sampling, random sampling, etc.

3) Extract $B_{O_{D2}}$ entities, relationships, or attributes from the knowledge graph.

4) Determine the number $A_{O_{D2}}$ of extracted entities, relationships, or attributes that match the ontology model with facts.

5) Calculate the ratio of $A_{O_{D2}}$ to $B_{O_{D2}}$ in the extracted results.

NOTE—If there are multiple ontology models, sample from multiple ontologies and conduct evaluations.

8.1.3 Ontology model consistency

The testing steps for this feature include the following:

a) Open the relevant knowledge modeling modules.

b) Examine all B_{O_E} entity types, relationships, or attributes in the constructed ontology model.

c) Filter out the corresponding raw data.

d) Determine the number of semantically consistent entity types, relationships, or attributes A_{O_E} in the extracted results.

e) Calculate the ratio of A_{O_E} to B_{O_E} in the extracted results.

8.1.4 Ontology model monotonicity

The testing steps for this feature include the following:

a) Open the relevant knowledge modeling modules.

b) Randomly select B_{O_M} entity types from the constructed ontology model.

c) Filter out the subtypes from the results of step b.

d) Determine the total number A_{O_M} of entities and relationships in the union of step b and step c.

e) Calculate the ratio of A_{O_M} to B_{O_M} in the extracted results.

8.1.5 Ontology model scalability

The testing steps for this feature include the following:

a) Open the relevant knowledge modeling modules.

b) Randomly select B_{O_M} entity types, relationship types, or attributes from the constructed ontology model.

c) Add, delete, or modify entity types, relationship types, or attributes A_{O_M} in the results of step b.

d) Search and screen whether the corresponding entities, attributes, relationships, etc., have been synchronized with the modifications.

e) Calculate the number of entity types, relationship types, and attributes C_{O_M} that can be synchronized with modifications in A_{O_M}.

f) Calculate the ratio of C_{O_M} to B_{O_M} in the extracted results.

8.2 Knowledge acquisition performance testing method

8.2.1 Structured data

The testing steps for the performance of knowledge acquisition modules include the following:

a) Open the relevant knowledge acquisition modules.

b) Configure the parameters for knowledge acquisition.

c) Record the start time of issuing commands.

d) Record the end time of issuing commands.

e) Count the number of entities, relationships, or attributes (TPA) acquired that match the ground truth.

f) Count the number of entities, relationships, or attributes (FPA) acquired that do not match the ground truth.

g) Evaluate completeness, i.e., check if the number of acquired entities, relationships, or attributes that match the ground truth is consistent.

h) Calculate accuracy (TPA / (TPA + FPA)) and acquisition speed per second.

i) Evaluate stability, i.e., repeat steps a-h multiple times and measure the fluctuation of parameters related to completeness and accuracy.

8.2.2 Semi-structured data

The testing steps for the performance of knowledge acquisition modules include the following:

a) Open the relevant knowledge acquisition modules.

b) Configure the parameters for knowledge acquisition.

c) Annotate the content of the target extraction text.

d) Record the start time of issuing commands.

e) Record the end time of issuing commands.

f) Filter the corresponding raw data.

g) Count the number of entities, relationships, or attributes (TPA) recognized that match the ground truth.

h) Count the number of entities, relationships, or attributes (FPA) recognized that do not match the ground truth.

i) Count the number of entities, relationships, or attributes (FNA) marked as ground truth but not recognized.

j) Calculate precision, recall, F1 score, acquisition speed per second, and data parsing speed.

k) Confirm whether relationship constraint validation and attribute constraint validation can be performed.

8.2.3 Unstructured data

The testing steps for the performance of knowledge acquisition modules include the following:

a) Open the relevant knowledge acquisition modules.

b) Configure the parameters for knowledge acquisition.

c) Annotate the content of the target extraction text.

d) Record the start time of issuing commands.

e) Record the end time of issuing commands.

f) Filter the corresponding raw data.

g) Count the number of entities, relationships, or attributes (TPA) recognized that match the ground truth.

h) Count the number of entities, relationships, or attributes (FPA) recognized that do not match the ground truth.

i) Count the number of entities, relationships, or attributes (FNA) marked as ground truth but not recognized.

j) Calculate precision, recall, F1 score, acquisition speed per second, and data processing speed (such as text analysis, audio, and video processing).

k) Confirm whether relationship constraint validation and attribute constraint validation can be performed.

8.3 Performance testing methods for knowledge storage

8.3.1 Information density of knowledge storage

The testing steps for measuring the information density of knowledge storage include the following:

a) Open the relevant knowledge storage modules.

b) Prepare the data set(s) for intended storage (e.g., multiple data sets for calculating averages) and determine the storage space occupied by the data set(s) (OPA).

c) Record the time when the command is issued.

d) Record the time when storage is completed.

e) Calculate the data storage time.

f) Count the number of entities, relationships, or attributes (TPA) stored as knowledge.

g) Calculate information density, TPA/OPA, which represents the quantity of storage per unit of storage space.

8.3.2 Data loading time

The testing steps for measuring data loading time include the following:

a) Open the relevant knowledge storage modules.

b) Prepare the data to be loaded and calculate the data volume.

c) Record the time when the command is issued.

d) Record the time when loading is completed.

e) Perform random sampling to evaluate the correctness of the loaded data.

f) Calculate the data loading time.

g) Calculate the loading time per unit data volume, measured in MB/s.

h) Execute steps c) to e) N times and calculate the average.

8.3.3 Response Time for k-hop neighbor queries

The testing steps for measuring the response time of K-hop neighbor queries include the following:

a) Select a uniform sample of 100 K-hop queries.

b) Open the relevant knowledge storage modules.

c) Configure the K-hop query parameters, such as query direction (both), the number of neighbors within K hops, starting node, K value, directional or undirectional, and the set of relationships or attributes.

d) Record the time when the command is issued.

e) Record the time when the query is completed.

f) Calculate the average query response time.

g) Perform random sampling to evaluate the correctness of the results.

h) Execute step c) to step e) N times and calculate the average.

8.3.4 Maximum concurrent queries supported for K-hop neighbor queries

The testing steps for measuring the maximum concurrent queries supported for K-hop neighbor queries include the following:

a) Open the relevant knowledge storage modules.

b) Set up a sequence of concurrent request numbers.

c) Using binary search, select the concurrent request number to be tested.

d) For the selected concurrent request number, conduct a concurrent testing.

e) For each request, test the response time or throughput of concurrent requests.

f) Monitor the test results and stop the test if:

   1) Abnormalities occur during the test, and the test cannot be completed normally.

   2) Response times become untestable.

g) If the maximum concurrent request number is not reached after testing, increase the number of attempts, or adjust the testing range and increment.

8.3.5 Throughput

The testing steps for measuring throughput include the following:

a) Determine point samples and edge samples.

b) Open the relevant knowledge storage modules.

c) Configure the number of concurrent request threads and the total number of operations.

d) Set up a sequence of concurrent operations.

e) Each thread iteratively performs the operation sequence.

f) Monitor the test results and stop the test when the cumulative number of operations equals the total number set in b).

g) For each request from each thread, evaluate the correctness of the results and record the response time of the request.

h) Calculate the average response time and throughput.

i) Additional stress tests can be performed sequentially, and throughput can be calculated for the following:

    1) Query point and query edge stress tests.

    2) Add point and add edge stress tests.

    3) Modify point and modify edge stress tests.

    4) Delete point and delete edge stress tests.

8.3.6 Queries per second (QPS) (single command)

The testing steps for measuring QPS (single command) include the following:

a) Open the relevant knowledge storage modules.

b) Configure the number of concurrent request threads and the duration.

c) Set the parameters for a single command query.

d) Each thread iteratively executes the set single command.

e) Monitor the test time and stop the test when the test time equals the duration set in b).

f) For each request from each thread, evaluate the correctness of the results and record the response time of the request.

g) Calculate QPS.

8.3.7 Queries per second (QPS) (multiple commands)

The testing steps for measuring QPS (multiple commands) include the following:

a) Open the relevant knowledge storage modules.

b) Configure the number of concurrent request threads and the duration.

c) Set up a sequence of parameters for multiple command queries.

d) Each thread iteratively executes the sequence of multiple commands.

e) Monitor the test time and stop the test when the test time equals the duration set in b).

f) For each request from each thread, evaluate the correctness of the results and record the response time of the request.

g) Calculate QPS.

8.3.8 Top-k hit ratio

The testing steps for measuring the top-k hit ratio include the following:

a) Open the relevant knowledge storage modules.

b) Set up K-hop query parameters.

c) Continuously send requests without interruption for a specified period and calculate the total number of requests.

d) Count the number of accurate queries.

e) Calculate the query hit ratio.

8.4 Performance testing methods for knowledge fusion

The performance testing steps for modules related to knowledge fusion include the following:

a) Open modules related to knowledge fusion.

b) Set parameters for knowledge fusion.

c) Count the number TPF of entities, relationships, or attributes in the fusion results that match the truth.

d) Count the number FPF of entities, relationships, or attributes in the fusion results that have been fused but do not match the truth.

e) Count the number FNF of entities, relationships, or attributes to be fused in the fusion results.

f) Count the accuracy, recall and F1 score.

8.5 Performance testing methods for knowledge computing

8.5.1 Response time

The testing steps include the following:

a) Open modules related to knowledge computing.

b) Prepare parameters related to knowledge calculation.

c) Execute knowledge reasoning, knowledge completion, and other related algorithms separately.

NOTE—Annex C shows the calculation methods of closeness centrality and betweenness centrality.

d) Count the time when instructions are issued.

e) Count the end time of loading.

f) Calculate the data loading time.

g) Execute d) to f) for N times and calculate the average value.

8.5.2 Accuracy

The testing steps include the following:

a) Open modules related to knowledge computing.

b) Prepare parameters related to knowledge calculation.

c) Continuously send knowledge inference, knowledge completion, and other related algorithms calculation requests for a period of time, and calculate the total number of requests.

d) Count the number of requests that meet expectations.

e) Calculate the number of correct requests/calculate the total number of requests.

8.5.3 Consumption of hardware resource

The testing steps include the following:

a) Open modules related to knowledge computing.

b) Prepare parameters related to knowledge calculation.

c) Continuously send knowledge inference, knowledge completion, and other related algorithms calculation requests for a period of time.

d) Observe CPU and memory usage rate.

e) Calculate the average value of CPU and memory usage rate.

8.6 Performance testing methods for knowledge provenance

8.6.1 Accuracy

The testing steps include the following:

a) Open modules related to knowledge provenance.

b) Randomly extract some data from knowledge and calculate the data volume(giving algebraic symbols).

c) Filter out the original data.

d) Determine the quantity of accurate knowledge(referring to knowledge with clear source, reliable, time-sensitive, and trusted content, and record of source) in the extracted results.

e) Calculate the proportion of d)/b).

f) Execute b) to e) N times and calculate the average value (provide the calculation formula).

8.6.2 Integrity

The testing steps include the following:

a) Open modules related to knowledge provenance.

b) Randomly extract some data from knowledge and calculate the data volume.

c) Filter out the original data.

d) Determine the quantity of knowledge with complete sources and transformation process(complete, traceable) in the extracted results.

e) Calculate the proportion of d)/b).

f) Execute b) to e) N times and calculate the average value.

8.6.3 Confidence level of traceability results

The testing steps include the following:

a) Open modules related to knowledge provenance.

b) Randomly extract some data from knowledge and calculate the data volume.

c) Filter out the original data.

d) Determine the quantity of complete and correct knowledge in the extracted results.

e) Calculate the proportion of d)/b).

f) Execute c) to e) N times and calculate the average value.

8.7 Performance testing methods for knowledge visualization

8.7.1 Rendering speed

The testing steps of rendering speed include the following:

a) Open modules related to knowledge representation.

b) Set the number of entities or relationships to be presented.

c) Count the time when requests are made.

d) Count the end time of rendering the statistical results.

e) Calculate rendering time T_p.

f) Calculate rendering speed X_p (corresponding to the calculation formula).

8.7.2 Maximum number of rendering nodes

The testing steps for the maximum number of rendering nodes include the following:

a) Open modules related to the knowledge representation. (Turn off attribute filtering.)

b) Set the number of entities or relationships to be presented.

c) Count the number of nodes at the end of the presentation.

d) Increase the number of entities or relationships to be presented.

e) Continue to execute b) until the number of nodes cannot continue to increase at the end of rendering (response time exceeds 30 s).

f) Count the number of nodes ultimately presented.

NOTE—Test duration: 10 min, 30 min, 1 h, corresponding (set 10 min test time)

9. Performance testing methods for knowledge graph application related modules

9.1 Responsiveness

9.1.1 Knowledge graph scale

Knowledge graph scale-related testing steps shall include the following:

a) Knowledge graph volume
   1) Run the system and select the knowledge graph to be measured.
   2) Statistically calculate number of entities in the knowledge graph V_e.
   3) Statistically calculate number of relationships in the statistical knowledge graph V_r, including gentity relationships and attribute relationships.
   4) Calculate the sum of the number of entities and the number of relationships V.

b) Knowledge graph complexity
   1) Run the system and select the knowledge graph to be measured.
   2) Statistically calculate number of entity types in knowledge graph F_e.
   3) Statistically calculate number of relationship types in knowledge graph F_r.
   4) Statistically calculate attribute items in knowledge graph F_a.
   5) Calculates F: the sum of the number of entity types, the number of relationship types, and the property item.

9.1.2 Knowledge graph quality

Knowledge graph quality-related testing steps shall include the following:

a) Knowledge coverage
   1) Run the system and select the knowledge graph to be measured.
   2) Statistically calculate number of entities in authoritative evidence in the field of statistics, such as guidelines, standards, policies D_{aut}, etc.
   3) Evidence of statistical practice, such as localization experience, special processes, or abnormal processes D_{pra}, etc.
   4) Weighting parameters that regulate the global impact of authoritative evidence and practical evidence, α.
   5) Statistically calculate number of entities in the statistical knowledge base that appears in authoritative and practical evidence N_{ent}.
   6) Calculate the degree of coverage CD of knowledge contained in the domain of knowledge-based assisted decision-making systems. It is recommended to add formulas.

b) Knowledge reasoning accuracy

   The requirements of 6.1.2 shall be met. The average of all expert scores is taken as the final knowledge reasoning accuracy. The calculation formula is as follows:

$$F = \frac{\sum_{i=1}^{M} f_i}{M} \times 100\% \tag{48}$$

where

F is the accuracy of knowledge reasoning
f_i is the rating of each expert
M is the number of scoring experts it is recommended to increase the weight of experts

9.1.3 Responsiveness efficiency

Test steps for responsiveness effectiveness shall include the following:

a) Run the system and select the knowledge graph to be measured.

b) Select the task content and determine the supporting request execution parameters, such as multi-hop query and path calculation.

c) Start the test program, the test program shall have the function of recording the start and end time.

d) According to the requirements of 6.1.3, determine the running time of the program.

e) Record the total number of requests processed during the run A_r.

f) Record the time T taken to process the request.

g) Calculate the number of requests processed per unit of time.

9.1.4 Response accuracy

Test steps for response accuracy shall include the following:

a) Run the system and select the knowledge map to be tested; tend to the quality of the knowledge map; send the request; judge the quality of the knowledge result of the request—whether the content covered is complete, whether the knowledge is right, and how is its external expansion ability.

b) Statistically calculate true positive TPR: the number of responses that are identified and match the truth.

c) Statistically calculate false positive FPR: the number of responses that are identified but do not match the real one.

d) Calculate the ratio of results that match the request (e.g., fuzzy queries, vector-based semantic search, path calculations) to the total number of returned results Precision_R.

9.1.5 Real-time responses

Testing steps for responding to real-time performance shall include the following:

a) Run the system and select the knowledge graph to be measured.

b) Start the test program; the test program shall have the function of recording the number of responses and response time.

c) Record the number of times the request has been responded to within a limited time A_R.

d) Record the total number of requests made A.

e) Record real-time response time T.

f) Calculate the proportion of responses to requests sent by users within a limited time X_A.

9.2 Reliability

9.2.1 Usability

Testing steps for usability shall include the following:

a) Mean time of failure
   1) Run the system and select the knowledge graph to be measured.
   2) Record the total number of unavailable states time T_F.
   3) Record the number of failures N_F.
   4) Calculate the length of time X_R, the system is unavailable at the time of the failure.

b) SLA
   1) Mean time to repair.
   2) The number of controls for access rights.

9.2.2 Maturity

Maturity-related test indicators shall include the following:

a) Troubleshooting rate
   1) Run the system and select the knowledge graph to be measured.
   2) Record the number of faults corrected N_C.
   3) Record the number of failures that occurred in the test N_F.
   4) Calculate X_C, situations in which system failures can be eliminated and repaired.

b) Average time between failures
   1) Run the system and select the knowledge graph to be measured.
   2) Record system running time T_R.
   3) Record number of failures in the recording system N_F.
   4) Calculate the length of time the system is running between failures T_F.

c) Failure rate
   1) Run the system and select the knowledge graph to be measured.
   2) Record number of failures in the recording system N_F.
   3) Record system running time T_R.
   4) Calculate the average number of failures over a defined period of time N_{FA}.

9.2.3 Fault tolerance

The fault tolerance test procedure shall comply with the requirements of 6.3.3.

9.2.4 Recoverability

The testing steps for this feature include the following:

a) According to the deployment of the system for computer rooms, applications, and component instances, formulate high-availability test plans such as network interruption, machine downtime, and forced instance stopping.

b) According to the plan, implement high-availability emergency situations. Examine whether the service can be recovered under abnormal conditions and the time to recover T_R.

9.3 Transferability

Transferability is measured by the average installation time and installation success rate in different hardware and software environments, and the testing steps for this feature include the following:

a) Develop a system-level software and hardware configuration list according to the hardware (CPU architecture, manufacturer brand, etc.) and operating system that needs to be supported.

b) Develop software lists for relational databases, graph databases, and non-relational databases according to the relevant components used in the schema.

c) In the list, combine the solutions to be tested according to the feasibility, and follow the instructions to perform steps d) through j) one by one.

d) Install and run the knowledge graph construction and application system to be tested, and record the installation time T.

e) Import the pre-designed graph model and data into the knowledge graph platform.

f) Test all interfaces of the application (including HTTP, RPC), etc., and record the test results.

g) Test all features of the entire app and document the test results.

h) If the data is imported and all interfaces and functions (including import and export, etc.) are tested and passed, the test passes under the condition of marking the software and hardware.

i) Calculate the average installation time.

j) According to 6.4, calculate Equation (28) to calculate the installation success rate.

9.4 Security

Security mainly includes the security of permissions, the security of stored content and the security of interactive information, and the test steps of this feature include the following:

a) Install and run the knowledge graph to be tested to build and apply the system, and import the corresponding test data set.

b) According to the application system instructions, check whether the definition of roles and users is supported and set up.

c) Use different users and roles to change data, illegally tamper, execute related function points, and other actions on the system. Examine whether the integrity requirements in Clause 6 are met.

d) Use unauthorized use, unauthorized use of interfaces, illegal tampering with data, sample attacks, data poisoning, etc., to investigate whether the requirements of controllability can be met.

e) After the corresponding settings, confirm whether the confidentiality requirements are met by capturing packets on the network and directly viewing the stored data.

f) By viewing unauthorized data, examine the privacy protection of data acquisition and interaction. Confirm that privacy requirements are met.

g) From the perspective of scheme and actual operation, examine whether the ability to review the operation actions and content of the map is available. Confirm that the requirements for reviewability are met.

Annex A

(informative)

Test process of knowledge graph construction and application system

The test process of knowledge graph construction and application system is divided into the following six stages and shown in Figure A.1:

— Requirement collection: Combined with the specific application scenario of the knowledge graph construction and application system, survey and sort out the specific requirements and conditions of the system to be tested in this application scenario, and form a requirement document.

— Requirement analysis: According to the requirement document, set the relevant parameters and index requirements in combination with the specific application scenario to form a parameterized test requirement document, and determine the test basis and test cycle according to the above document and the corresponding national standards and industry standards.

— Test plan: According to the test requirement document and the corresponding standards, develop a test plan. The test plan shall specify the allocation of resources such as personnel, software, hardware and sample data environment, the test scheduling, the risk identification in the test process, and the standards for the success and failure of the test.

— Test design: According to the requirement document and the test plan, design the test cases, clarify the test strategy, determine the test method of the system to be tested, design the test process, build the test environment, and thereafter select the industry samples under specific application scenarios.

— Test execution: According to the determined test scope, objectives and test process, select and execute the test cases, select test mode such as unit-level test and system-level test, and test the system to be tested.

— Test summary: Issue the test report according to the test outcomes and evaluate whether the test meets the requirements of the corresponding scenarios.

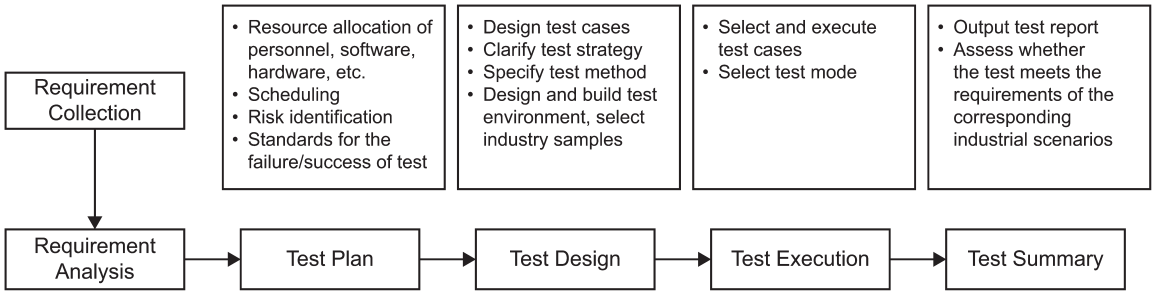

Figure A.1—Test process of knowledge graph construction and application system

Annex B

(informative)

Test cases

B.1 Test case of data loading time

Test number	001
Test item	Data loading time
Test purpose	Test the efficiency of importing external data to graph database.
Test environment	The test uses a standard test environment: server model, number of servers AA,. memory BB GB RAM, number of server CPUs CC, operating system DD The test uses a standard test data set: data set name, data name EE, vertex number FF, edge number GG, file size HH GB.
Preconditions	1. Open the graph database management system and log in. 2. Prepare the data set to be imported. 3. To guarantee the fairness of the test, the officially provided import tool is adopted for the test, and the corresponding tools are prepared. 4. Considering the fact that common data sets have provided the edge files and the vertex files, the time to obtain the vertex files through the preprocessing of the edge files is not counted.
Test procedures	1. Configure the data sets that need to be imported. 2. Read in the edge data, prepare the vertex files, read in the vertex data, and build the indexes. 3. Record the execution time of the task.
Expected outcome	The operation of data import succeeds, and the data is correct.
Test outcome	Calculate the data loading time.

B.2 Test case of response time of K-hop neighbor query

Test number	002
Test item	Response time of K-hop neighbor query.
Test purpose	Test the performance of neighbor query operation.
Test environment	The test uses a standard test environment: server model, number of servers 1, memory 64 GB RAM, number of server CPUs 32, a common operating system. The test uses a standard test data set: data set name, data name AA, vertex number BB, edge number CC, file size DD GB.
Preconditions	1. Open the graph database management system and log in. 2. Random seeds are required in the test of K-hop data query. Count the average number of K-hop neighbors starting from the random seed. 3. Test with algorithms such as Latency.
Test procedures	1. Use the original query language to query the K-hop neighbors of M random seed and set the query timeout. 2. Query the data of the K-hop neighbors for each vertex and record the execution time and the timeout period of the query task. 3. Calculate the average query time and timeout period.
Expected outcome	The result of K-hop query is correct.
Test outcome	Calculate the average query time and timeout period of N random seeds for K-hop queries.

Annex C

(informative)

Calculation methods of closeness centrality and betweenness centrality

The calculation methods of closeness centrality and betweenness centrality are described as follows:

Closeness centrality measures the difficulty of a specific node in the knowledge graph to reach the other nodes. The calculation formula is:

$$C(u) = \frac{n-1}{\sum d(u,v)} \tag{C.1}$$

where

u is a specific entity
n is the number of entities in the knowledge graph
$d(u,v)$ is the distance of the shortest path between entities v and u

Betweenness centrality measures the number of times an entity is part of the shortest path between two other entities. The calculation formula is:

$$B(u) = \sum_{s \neq u \neq t} \frac{p(u)}{p} \tag{C.2}$$

where

u is a specific entity
p is the number of shortest paths between entities s and t
$p(u)$ is the number of shortest paths through u between s and t

IEEE SA
STANDARDS ASSOCIATION

RAISING THE WORLD'S STANDARDS

Connect with us on:

Facebook: facebook.com/ieeesa
LinkedIn: linkedin.com/groups/1791118
Beyond Standards blog: beyondstandards.ieee.org
YouTube: youtube.com/ieeesa

standards.ieee.org
Phone: +1 732 981 0060

IEEE